Yale University
Department of Computer Science

Efficient Retiming under a General Delay Model

Kumar N. Lalgudi[1] Marios C. Papaefthymiou[2]

YALEU/DCS/RR-1048
September 1994

[1]Department of Electrical Engineering
[2]Department of Electrical Engineering and Department of Computer Science

Efficient Retiming of Edge-Triggered Circuits under a General Delay Model

Kumar N. Lalgudi Marios C. Papaefthymiou

Department of Electrical Engineering
Yale University
New Haven, CT 06520

Abstract

The retiming transformation can be used to optimize synchronous circuits for maximum speed of operation by relocating their storage elements. Existing retiming algorithms that run in polynomial time work with relatively simple delay models that neglect several timing issues that arise in logic design. Recent retiming algorithms for more comprehensive delay models rely on non-linear formulations and run in worst-case exponential time using branch-and-bound techniques. In this paper, we investigate the retiming problem for edge-triggered circuits under a general delay model that handles load-dependent gate delays, register delays, interconnect delays, and clock skew. We show that in this model the retiming problem can be expressed as a set of integer linear programming (ILP) constraints that can be solved using general ILP techniques.

For the special case where clock skew is monotonic and all registers have equal propagation delays, we give an integer monotonic programming formulation of the retiming problem, and we present an efficient algorithm for solving it. Our algorithm retimes any given edge-triggered circuit to achieve a specified clock period in $O(V^3F)$ steps, where V is the number of logic gates in the circuit and F is bounded by the number of registers in the circuit. A straightforward extension of our algorithm determines a minimum clock period retiming in $O(V^3F \lg V)$ steps. We have implemented our retiming algorithms in an efficient and versatile tool called DELAY. We illustrate the effectiveness of our tool by presenting experimental results from its application on MCNC benchmark circuits.

1 Introduction

As minimum feature sizes continue to shrink, factors such as interconnect capacitance, clock skew, and fanout load are becoming increasingly significant in the total delay of a combinational path. Traditional architectural-level methods for optimizing synchronous circuits are applied at a stage where these factors are either indeterminate or very difficult to estimate, and consequently, they resort to simple delay models for timing analysis and optimization. When optimization techniques are applied at such an early stage, however, the timing characteristics of the resulting designs can change significantly after layout. This limitation can be overcome by optimizing designs at the post-placement stage of the design cycle. At this stage, the global routing information available can provide good estimates for interconnect delays, clock distribution delays and fanout loads. Therefore, post-placement optimization algorithms must be able to take these additional factors into consideration.

Retiming is a popular and powerful architectural-level transformation that optimizes synchronous circuits by relocating their storage elements without affecting their functionality. Existing retiming algorithms that run in polynomial time assume relatively simple delay models that neglect the timing effects of several factors such as load-dependent gate delays and clock skew [12, 13, 19]. The successful application of retiming in a post-placement optimization scenario requires the effective and efficient handling of general delay models.

However, recently proposed retiming algorithms that operate under comprehensive delay models rely on non-linear formulations of the problem that are solved using branch-and-bound techniques that run in worst-case exponential time [21, 22].

In this paper, we investigate the problem of retiming edge-triggered circuits for clock period optimization under the general delay model proposed in [21, 22]. This model is more general than the simple models assumed in polynomial-time algorithms and takes into account load-dependent gate delays, variable register setup times, hold times and propagation delays, interconnect delays, and clock skew. Our investigation has resulted in improved retiming algorithms which we have implemented in a software tool called DELAY. The efficient operation of DELAY relies on a novel integer linear programming formulation of the retiming problem for satisfying both setup and hold time constraints in this model. For a restricted version of the general model, which includes the case where clock skew is monotonic and all registers have equal propagation delays, our retiming algorithm satisfies setup time constraints and runs in polynomial time. Specifically, our algorithm can retime any given edge-triggered circuit to achieve a specified clock period, or report that the specified clock period cannot be achieved by retiming the original circuit, in $O(V^3F)$ steps, where V is the number of logic gates in the circuit and F is bounded by the number of registers in the circuit. The efficient operation of this algorithm is based on a powerful integer programming technique called *monotonic programming*. A straightforward extension of our algorithm can determine a minimum clock period retiming in $O(V^3F \lg V)$ steps. We demonstrate the effectiveness of our schemes by presenting an empirical comparison of the accuracy and running times of the algorithms in DELAY when applied on a subset of the MCNC benchmark circuits. Our experimental results indicate that the retiming algorithm based on monotonic programming runs almost as efficiently as the algorithms that work with simple delay models. Moreover, the monotonic programming algorithm yields solutions that approach those obtained with our exact integer linear programming scheme for the general delay model.

The remainder of this paper has six sections. In Section 2, we provide a brief summary of existing literature related to our paper. In Section 3 we illustrate the intricacies of retiming under the general delay model we consider, and in Section 4 we describe the model and its graph representation in more detail. In Section 5 we give an integer linear programming formulation of the retiming problem in this model. In Section 6 we describe the integer monotonic programming formulation of retiming in the special case of monotonic clock skew and equal register delays. We present our polynomial-time algorithm for solving this problem in Section 7. In Section 8 we discuss our experimental results obtained by applying DELAY on a collection of test circuits. We conclude in Section 9 with directions for further research.

2 Previous Work

In this section, we review existing literature on retiming and summarize previous efforts to retime synchronous circuits under more general delay models.

Retiming has proved to be a successful technique for optimizing area, clock speed, testability and low power [2, 6, 16, 17, 20]. Retiming (or " firing") was first introduced by Commoner *et. al.* [4] as a technique for state minimization in "marked graphs". A systematic treatment of retiming for optimizing the operating speed of synchronous circuits was given by Leiserson and Saxe in [13]. The authors of that paper studied retiming under a simple delay model in which every circuit component has a fixed delay and the propagation delay

of any given path increases monotonically with the number of its components. Based on the properties of this model, they formulated the retiming problem as a set of shortest-paths constraints that can be solved using an $O(V^3 \lg V)$-time Bellman-Ford algorithm. They also described an asymptotically faster retiming algorithm that runs in $O(VE \lg V)$ steps. Papaefthymiou [19] has given bounds for the optimal clock period that can be achieved by retiming and has described an $O(V^{1/2}E \lg V)$-time algorithm for optimal retiming of unit-delay circuits, which is the asymptotically fastest known algorithm for the problem.

The clock speed optimization algorithms that use the simple delay model proposed in [13] rely on the concept of *path-breaking*: whenever the delay of a combinational path exceeds the desired clock period, then the path is broken by introducing a register somewhere in between. However, path-breaking can result in sub-optimal designs under a more general delay model, as we argue in Section 3. This limitation is intrinsic to the delay monotonicity assumed by the path-breaking algorithms and is not simply a consequence of the fact that these schemes do not use comprehensive information about delays in the circuit. As we argue in Section 8, the path-breaking algorithms will generate sub-optimal designs even when they compute path delays using general delay models. For example, the modified version of the $O(V^3 \lg V)$-time Bellman-Ford retiming algorithm generates designs with sub-optimal clock periods, while the modified version of the $O(VE \lg V)$-time algorithm can result in invalid circuits.

Several researchers have studied retiming in conjunction with clock skew. Fishburn has drawn attention to the fact that any clock period that can be achieved by retiming a circuit can also be achieved by adjusting the skew of its clock lines [7]. Chao and Sha have proposed a two-step procedure to optimize synchronous circuits for clock speed by combining retiming with clock skew adjustment [3]. The first step of their scheme retimes a unit-delay version of any given circuit to achieve maximum speed of operation. Subsequently, storage elements are shifted out of combinational blocks that cannot be broken, and clock skew is adjusted to maintain clock period. The potential advantage of this approach over the general scheme presented in [7] is that for the same clock period, it may result in smaller clock skew adjustments.

Lockyear and Ebeling have considered the effect of monotonic clock skew on retiming and circuit performance of level-clocked circuits [15]. These researchers have shown that the paths between vertex pairs that dictate the clock period under a simple delay model do not change in the presence of monotonic clock skew and constant latch setup times. They have thus concluded that the form of retiming constraints does not change and that existing retiming algorithms such as those in [9, 14] can still be applied. As we discuss in this paper, however, for more comprehensive delay models that include general clock skew, load-dependent delays, and variable register delays, existing retiming algorithms are not suitable.

The research most relevant to our work is that of Soyata, Friedman, and Mulligan who have studied the problem of incorporating clock skew, variable register delay and interconnect delay into the retiming process [21, 22]. These researchers have expressed the retiming problem as a set of non-linear constraints that are solved using general branch-and-bound techniques with a worst-case exponential running time. In this paper, we give an integer linear programming formulation of the retiming problem for setup and hold constraints (that is, long and short paths) under the general delay model proposed in [21, 22]. Our formulation enables the use of ILP solvers which typically use efficient linear programming code to speed up the branch-and-bound search. In these solvers, the integer bounds of the variables in each iteration are determined by rounding off the solution to

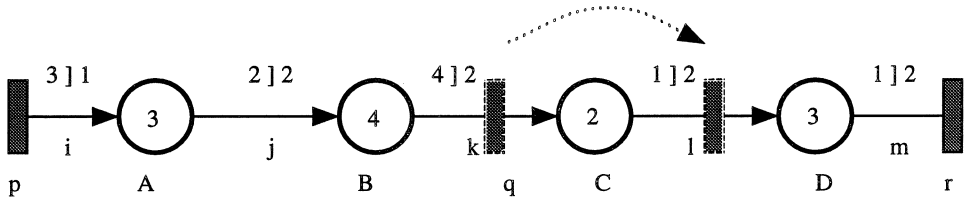

Figure 1: Intricacies of retiming in a general model.

a linear programming relaxation of the original problem. In addition to the general ILP formulation, we give an efficient retiming algorithm for satisfying setup constraints under a restricted yet interesting version of the general delay model. The running time of this algorithm is polynomial in the size of the input circuit.

3 Intricacies of Retiming

The correctness and computational efficiency of the early retiming algorithms has relied on the key property of *path delay monotonicity* which ensures that propagation delays along combinational paths increase monotonically as the number of logic gates on the paths increases. Path delay monotonicity provides a straightforward way to satisfy timing constraints. Whenever the delay of a combinational path exceeds the desired clock period, the only way to fix the timing violation is by pulling the registers in the beginning and at the end of the path closer to each other. The monotonicity of propagation delays along paths is a consequence of the simple assumptions that have been made in the delay models used by the early algorithms.

The main difficulty with retiming in a general delay model such as the one we consider in this paper stems from the fact that path delays are not necessarily monotonic. Due to this lack of monotonicity, critical delays may decrease when combinational paths are extended and increase when registers are brought closer together. Thus, it may be possible to meet a timing constraint along a combinational path by pulling away its bounding registers as opposed to "breaking" the path by moving a register into it. For example, when a register at the output of a gate is shifted farther down the circuit, the propagation delay through that gate will decrease if the input capacitance of the register exceeds that of the logic that is seen from the gate's output after the register's relocation. This situation can occur when designing with standard cells such as the ones in the CMOS3 cell library [8]. As a result, the propagation delays of paths can be reduced by extending the paths to include additional logic gates. Another situation where path delays change in effect non-monotonically arises when we consider clock skew. When timing is violated along a combinational path, it may be advantageous to move its bounding registers farther apart, because the differences in the arrival times of the clocking signals to the bounding registers may allow more time for data to propagate along the combinational path enclosed between the new register locations. From a computational perspective, the main source of difficulty behind retiming with a general delay model is the *decision nature* of the problem. With simple delay models we know that registers always move in a direction that reduces the number of vertices along a combinational path. With more general models, when we relocate a register to achieve correct circuit timing, we must decide whether we should move to increase or decrease the number of vertices along a combinational path.

Figure 1 illustrates the intricacies that arise in retiming when we consider a general

model that takes into account the effects of load-dependent gate delays, clock skew, register setup times and propagation delays, and interconnect delays. The circuit shown in this figure has four blocks of combinational logic (A, B, C, and D) and three registers (p, q, and r). Each edge in the graph denotes a wire and is labeled with a pair of numbers $x]y$ that give the additional propagation delays incurred by signals when there is a register on that wire. Specifically, x gives the additional delay for all signals that end on this register, and y gives the additional delay for all signals that start from it. (The numbers x and y give the total effect of the various factors on propagation delay and may take on positive or negative values. In Section 4 we discuss how these numbers are obtained.) When register q is placed on wire k, the path $p \xrightarrow{i} A \xrightarrow{j} B \xrightarrow{k} q$ has a propagation delay of $1 + 3 + 4 + 4 = 12$ units of time. By shifting register q onto wire l, the combinational path is extended to $p \xrightarrow{i} A \xrightarrow{j} B \xrightarrow{k} C \xrightarrow{l} q$. The propagation delay along this extended path, however, decreases to $1 + 3 + 4 + 2 + 1 = 11$ units of time. Assuming that registers p and r cannot move, the only way to achieve a clock period of 11 is to increase the number of combinational logic blocks in the path instead of decreasing it.

4 Preliminaries

In this section, we describe the delay model that we consider and give a graph representation of an edge-triggered circuit in this model. We also provide some background on retiming, and show how to compute an $O(E^2)$-size set that is guaranteed to include the optimal clock period achievable by retiming any given circuit under our general delay model. Under a restricted version of our delay model where clock skew is monotonic and all register propagation delays are equal, we describe how to obtain an $O(VE)$-size set of potential clock periods for any given circuit.

4.1 Delay Model

The delay model that we consider in this paper adheres to the basics of the *Register Electric Characteristics* model that was proposed in [21, 22]. This model takes into account load-dependent gate delays, register setup times and propagation delays, clock skew, and interconnect delays. Specifically, for each wire i in the circuit, this model assumes a clock delay $t_C(i)$ for the propagation of the clock from a global clock source to any register on that wire, a delay $t_{C \to Q}(i)$ for the data to appear at the output of any register on i upon arrival of the clock signal, and a setup time $t_s(i)$ for any register on i. Moreover, this model assumes an interconnect delay $d_{i_1}(i, u)$ for the propagation of any signal from a register on the wire $u_i \xrightarrow{i} u$ to the block u, an interconnect delay $d_{i_2}(v, j)$ for the propagation of a signal from a block v to a register on the wire $v \xrightarrow{j} v_j$, and can be straightforwardly extended to include the delay of the interconnect between combinational logic blocks.

In our analysis, each block u is associated with a number $d(u)$ which gives its propagation delay when it sees a combinational block at its fanout. Moreover, each fanout wire $u \xrightarrow{j} v$ is associated with a number $d_l(u, j)$ which gives the change in the propagation delay of any signal that reaches j via u when a register is present on j. Depending on the relation between the capacitive loads presented by a register on j and by the fanout vertex v, the parameter $d_l(u, j)$ can take positive or negative values. In this paper, we assume that for any combinational path p, the propagation delay along p depends only on the elements on p. Thus, the worst-case delay $\Delta(i, p, j)$ of a combinational register-to-register path

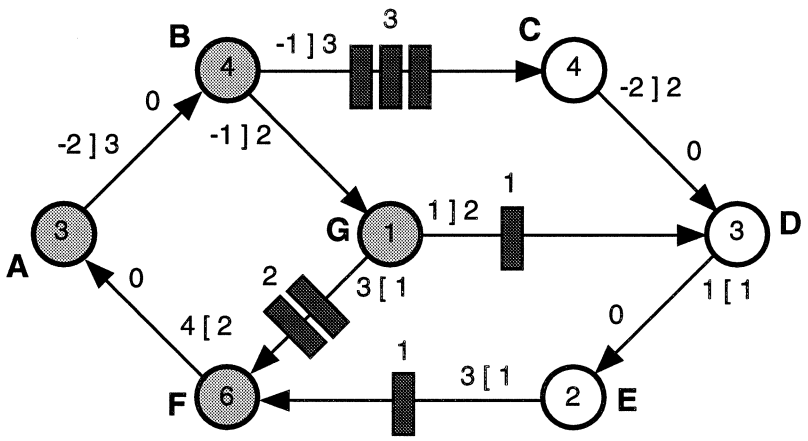

Figure 2: Graph representation of an edge-triggered circuit. The numbers in the vertices denote the propagation delay through the corresponding combinational block when the output of the block on the path we consider "sees" another combinational block. The numbers on the edges denote the delay values $t_e]t_b$ which handle the extra delays when registers are placed on these edges.

$u_i \xrightarrow{i} u \overset{p}{\rightsquigarrow} v \xrightarrow{j} v_j$, where u_i, u, v, v_j are logic blocks and i, j are wires with registers on them, is given by the equation

$$\Delta(i,p,j) \;=\; t_C(i) + t_{C\to Q}(i) + d_{i_1}(i,u) + \sum_{x\in p} d(x) + d_l(v,j) + d_{i_2}(v,j) + t_s(j) - t_C(j)\ .$$

4.2 Graph Representation

An edge-triggered circuit is modeled as a directed multigraph $G = \langle V, E, d, w, t_b, t_e\rangle$. The vertices V in the graph correspond to the combinational elements in the circuit. Each vertex $v \in V$ is associated with a weight $d(v)$ which gives the propagation delay through the block when it "sees" another combinational block down the path. The directed edges E of the graph model the interconnections between the combinational blocks. Each edge $e \in E$ corresponds to a wire that connects an output of some combinational block to the input of another block, and it is associated with a weight $w(e)$ that gives the register count on that wire.

Each edge $e \in E$ is also associated with weights $t_b(e)$ and $t_e(e)$ which account for the changes in propagation delays when registers are present on e. For each edge $u \xrightarrow{i} v$, the weights $t_b(i)$ and $t_e(i)$ are equal to

$$\begin{aligned} t_b(i) &= t_C(i) + t_{C\to Q}(i) + d_{i_1}(i,v)\ , \\ t_e(i) &= d_l(u,i) + d_{i_2}(u,i) + t_s(i) - t_C(i)\ , \end{aligned}$$

and give the extra delay incurred by signals whose propagation begins or ends at a register on i, respectively. Thus, the effective delay of the combinational register-to-register path $u_i \xrightarrow{i} u \overset{p}{\rightsquigarrow} v \xrightarrow{j} v_j$ is given by the equation

$$\Delta(i,p,j) = t_b(i) + \sum_{x\in p} d(x) + t_e(j)\ .$$

Figure 2 illustrates the graph representation of a synchronous circuit. The longest combinational path in that circuit is denoted by the shaded blocks and its propagation delay is 18. Thus the circuit requires a minimum clock period of 18 units of time.

The difficulty of the retiming problem depends on the values of the edge-weights t_b and t_e. When these weights take any real value, it may be possible to satisfy timing constraints by extending paths in either of the two directions possible. In this case, we call the corresponding circuits *two-way extendible*. When all weights t_b are nonnegative, it may be possible to fix a timing violation along a path by extending the end of the path. We call the circuits with this property *one-way end-extendible*. Symmetrically, we call *one-way begin-extendible* the circuits in which all weights t_e are nonnegative, because in these circuits it may be possible to fix timing violations by extending the beginning of a violating path.

Based on the definitions of t_b and t_e, we can associate circuit properties with circuit extendibility. For example, it is straightforward to verify that when clock skew behaves monotonically along every path p, that is, $\sum_{x \in p} d(x) + d_j(v) \geq t_C(j) - t_C(i)$, and all register propagation delays $t_{C \to Q}(i)$ are equal, then the circuit is one-way end-extendible.

4.3 Retiming

A retiming of an edge-triggered circuit $G = \langle V, E, d, w, t_b, t_e \rangle$ is an integer valued vertex-labeling $r : V \to \mathbf{Z}$. This labeling denotes the assignment of a lag to each vertex which transforms G into $G_r = \langle V, E, d, w_r, t_b, t_e \rangle$, where for each edge $u \xrightarrow{e} v$ in G, w_r is defined by the equation

$$w_r(e) = w(e) + r(v) - r(u) \; . \tag{1}$$

In order for G_r to be *well-formed*, the retiming r must satisfy the constraint $w_r(e) \geq 0$ for all edges $e \in E$. By adding Equation (1) along a path $u \overset{p}{\rightsquigarrow} v$, it can be shown that

$$w_r(p) = w(p) + r(v) - r(u) \; , \tag{2}$$

where $w_r(p) = \sum_{e \in p} w_r(e)$ and $w(p) = \sum_{e \in p} w(e)$.

An important circuit parameter in the context of retiming is the delay $D_{total}(i, j)$. For every pair of edges $u_i \xrightarrow{i} u$ and $v \xrightarrow{j} v_j$ in G, this delay is defined as

$$D_{total}(i, j) = t_b(i) + D(u, v) + t_e(j) \; ,$$

where $D(u, v) = \max\{\sum_{x \in p} d(x) : u \overset{p}{\rightsquigarrow} v, \text{ and } w(p) = W(u, v)\}$, $w(p)$ denotes the register count of a path p, and $W(u, v) = \min\left\{w(p) : u \overset{p}{\rightsquigarrow} v\right\}$. The quantity $D(u, v)$ is the maximum combinational delay along the paths between the vertices $u, v \in V$ with the fewest registers. Since Equation (2) holds for all paths between every vertex pair (u, v), it can be shown that in the retimed circuit G_r, we have

$$W_r(u, v) = W(u, v) + r(v) - r(u) \; . \tag{3}$$

Another important parameter is the delay $\delta_{total}(i, j)$ which is defined for every pair of edges $u_i \xrightarrow{i} u$ and $v \xrightarrow{j} v_j$ in G as

$$\delta_{total}(i, j) = t_b(i) + \delta(u, v) + d_l(v, j) + d_{i_2}(v, j) - t_C(j) - t_h(j) \; ,$$

where $t_h(j)$ denotes the hold-time requirement on $v \xrightarrow{j} v_j$ and $\delta(u, v) = \min\{\sum_{x \in p} d(x) : u \overset{p}{\rightsquigarrow} v, \text{ and } w(p) = W(u, v)\}$. The quantity $\delta(u, v)$ is the minimum combinational delay along the paths between the vertices $u, v \in V$ with the fewest registers. The quantity $\delta_{total}(i, j)$ must be non-negative to satisfy the hold-time requirements of the register on edge j; otherwise the circuit G can have race conditions.

Since there are $O(E^2)$ pairs of edges, the parameters $D_{total}(i,j)$ and $\delta_{total}(i,j)$ can take on $O(E^2)$ values. Since the $O(V^2)$ delays $D(u,v)$ or $\delta(u,v)$ can be computed in $O(VE + V^2 \lg V)$ time [13], the $O(E^2)$ values $D_{total}(i,j)$ or $\delta_{total}(i,j)$ can be computed in $O(VE + V^2 \lg V) + O(E^2) = O(E^2 + V^2 \lg V)$ steps. The following lemma shows that the parameters $D_{total}(i,j)$ can be used to compute a set of values that is guaranteed to include the minimum clock period achievable by retiming a given circuit.

Lemma 1 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be an edge-triggered circuit, and let r be a retiming of G. Then the minimum clock period $\Phi_{\min}(G_r)$ that can be achieved by retiming G equals $D_{total}(i,j)$ for some pair of edges $i, j \in E$.*

Proof. The clock period of an edge-triggered circuit is determined by the effectively longest register-to-register combinational path in the circuit. Since the values $D_{total}(i,j)$ denote the worst-case delays of every potential register-to-register path in the retimed circuit, some $D_{total}(i,j)$ will equal the optimal clock period $\Phi_{\min}(G_r)$. □

Since there are $O(E^2)$ possible edge pairs, the size of the set of values that includes the optimal clock period is $O(E^2)$. However, for circuits whose paths are only end-extendible, the clock period for the latch-to-latch path $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ is dictated by the worst-case fanin edge of u. Thus, the number of potential clock periods decreases from $O(E^2)$ down to $O(VE)$. By a symmetric argument, we can show that this bound also holds for begin-extendible circuits. Moreover, it is straightforward to prove that in circuits whose fanin and fanout edges exhibit identical delay behavior, that is, for any given vertex u in the circuit, the parameters t_b of u's fanin edges are equal and the parameters t_e of u's fanout edges are equal, there are only $O(V^2)$ potential clock periods.

5 Integer Linear Programming Formulation

In this section, we show that the retiming problem for two-way extendible circuits can be formulated as a set of $O(E^2)$ integer linear programming constraints. We begin by describing a non-linear set of necessary and sufficient conditions for a retiming to satisfy the setup and hold time constraints of any given circuit. We then describe a novel graph transformation of our circuit and give integer linear constraints on the transformed circuit which are equivalent to the non-linear constraints on the original circuit.

The following lemma gives necessary and sufficient conditions for the clock period $\Phi(G_r)$ of a retimed circuit G_r to meet a specified clock period c.

Lemma 2 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be an edge-triggered circuit, let c be a positive real number, and let $r : V \to \mathbf{Z}$. Then r is a retiming of G such that $\Phi(G_r) \leq c$ if and only if for every edge $u \xrightarrow{e} v$ in G, we have*

$$0 \leq w_r(e) \ , \tag{4}$$

and for every edge pairs $i, j \in E$ such that $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ and $D_{total}(i,j) > c$, we have

$$W_r(u,v) = 0 \ \Rightarrow \ w_r(i) = 0 \ \text{or} \ w_r(j) = 0 \ . \tag{5}$$

Proof. ($\Rightarrow$) Let G_r be a well-formed circuit such that $\Phi(G_r) \leq c$. We want to show that Inequality (4) and Relation (5) hold for G_r. Since G_r is well formed, all edge-weights w_r in G_r must satisfy $w_r \geq 0$, and thus from Equation (1), we infer that Inequality (4) holds.

Let us assume that Relation (5) does not hold for G_r. Then there exists an edge pair $u_i \xrightarrow{i} u, v \xrightarrow{j} v_j \in E$ with nonzero $w(i)$ and $w(j)$ and $D_{total}(i,j) > c$ such that $W_r(u,v) = 0$. Therefore, there exists a register-to-register combinational path $i \rightsquigarrow j$ in G_r whose delay exceeds the clock period c, thus contradicting our assumption that $\Phi(G_r) \leq c$.

($\Leftarrow$) Inequality (4) ensures that G_r is well-formed. Relation (5) ensures that for every edge pair i, j such that $D_{total}(i,j) > c$, the path $i \rightsquigarrow j$ in G_r is not a register-to-register combinational path. This condition ensures that the clock period constraint $\Phi(G_r) \leq c$ is satisfied. □

Lemma 2 gives the conditions under which the setup time constraints are met by the retimed circuit for a given clock period. Similarly, we can state the necessary and sufficient conditions for the retimed circuit to be free from race conditions (hold time constraints) as follows.

Lemma 3 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be an edge-triggered circuit, and let $r : V \rightarrow \mathbf{Z}$. Then r is a retiming of G such that G_r is race-free if and only if for every edge $u \xrightarrow{e} v$ in G, we have*

$$0 \leq w_r(e) \ , \tag{6}$$

and for every edge pairs $i, j \in E$ such that $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ and $\delta_{total}(i,j) < 0$, we have

$$W_r(u,v) = 0 \ \Rightarrow \ w_r(i) = 0 \ \textit{or} \ w_r(j) = 0 \ . \tag{7}$$

Proof. Similar to Lemma 2. □

From Lemma 3, we conclude that setup and hold time constraints have the same form under the general delay model. For the remainder of this section, therefore, we restrict ourselves to setup time constraints as the treatment for hold time constraints is identical. Relation (5) cannot be directly expressed in a linear programming form because of the disjunction (*or* requirement) in the implication. To obtain a linear program, we construct a companion graph G' by segmenting every edge $u \xrightarrow{e} v \in E$ into two edges $u \xrightarrow{e_1} x_{uv}$ and $x_{uv} \xrightarrow{e_2} v$, where x_{uv} is a dummy vertex. Specifically, $G' = \langle V', E', w' \rangle$ is defined as

$$\begin{aligned} V' &= V \cup \left\{ x_{uv} : u \xrightarrow{e} v \in E \right\} \ , \\ E' &= \left\{ u \xrightarrow{e_1} x_{uv}, x_{uv} \xrightarrow{e_2} v : u \xrightarrow{e} v \in E \right\} \ , \end{aligned}$$

and for each edge $u \xrightarrow{e} v$ in E, we have

$$\begin{aligned} w'(e_1) &= \min\{1, w(e)\} \ , \text{ and} \\ w'(e_2) &= w(e) - \min\{1, w(e)\} \ . \end{aligned}$$

The following lemma recasts the conditions in Lemma 2 in terms of the companion graph G'.

Lemma 4 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be an edge-triggered circuit, and let $G' = \langle V', E', w' \rangle$ be its companion graph. Moreover, let c be an arbitrary positive real number. Then there exists a retiming $r : V \rightarrow \mathbf{Z}$ such that $\Phi(G_r) \leq c$ if and only if there exists a function $r' : V' \rightarrow \mathbf{Z}$ such that for every edge $u \xrightarrow{e_1} x_{uv}$ in E', we have*

$$0 \leq w'_{r'}(e_1) \ , \tag{8}$$

$$w'_{r'}(e_1) \leq 1 \ , \tag{9}$$

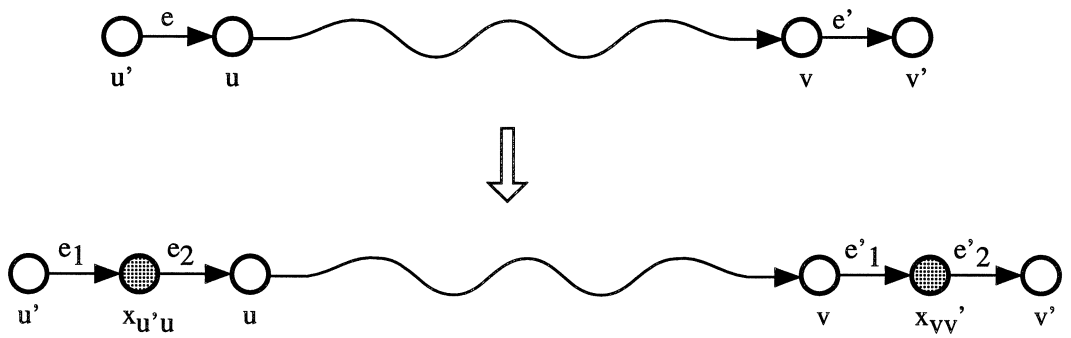

Figure 3: Generating a companion graph G' from a graph G. Every edge e in G is segmented into two edges e_1 and e_2 in G' such that e_1 contains at most one register and e_2 can contain a register only if there already exists a register on e_1. Constraints in G between a pair of edges e and e' of the form given in Relation (5) are transformed into an equivalent set of four constraints in G' involving pairwise segments of i and j of the form given in Relation (5).

for every edge $x_{uv} \xrightarrow{e_2} v$ *in* E', *we have*

$$0 \leq w'_{r'}(e_2) \ , \tag{10}$$

for every pair of edges $u \xrightarrow{e_1} x_{uv}$ *and* $x_{uv} \xrightarrow{e_2} v$ *in* E', *we have*

$$w'_{r'}(e_1) = 0 \ \Rightarrow \ w'_{r'}(e_2) = 0 \ , \tag{11}$$

and for every pair of edges $u' \xrightarrow{e_1} x_{u'u}, v \xrightarrow{e'_1} x_{vv'}$ *in* E' *such that* $u' \xrightarrow{e} u$, $v \xrightarrow{e'} v'$ *in* E, $u \rightsquigarrow v$ *in* G, *and* $D_{total}(e, e') > c$, *we have*

$$W'_{r'}(u, v) = 0 \ \Rightarrow \ w'_{r'}(e_1) = 0 \ \textit{or} \ w'_{r'}(e'_1) = 0 \ . \tag{12}$$

Proof. ($\Rightarrow$) Let G_r be a well-formed circuit such that $\Phi(G_r) \leq c$. We must show that constraints (8) through (12) are satisfied by a retiming r' of G'. Let r' be defined as follows

$$\begin{array}{lcll} r'(u) & = & r(u) & \text{for all } u \in V; \\ r'(x_{uv}) & = & r(u) & \text{for all } u \xrightarrow{e} v \in E \text{ such that } w_r(e) = 0; \\ r'(x_{uv}) & = & r(u) + 1 & \text{for all } u \xrightarrow{e} v \in E \text{ such that } w_r(e) \geq 1 \ . \end{array}$$

Since G_r is well-formed, we have that $w_r(e) \geq 0$, and therefore, Inequalities (8) and (10) are satisfied. From the definition of a companion graph, Inequality (9) and Relation (11) must be satisfied for the edge-weights in $G'_{r'}$. Since $\Phi(G_r) \leq c$, Relation (5) holds for G_r. Now, we observe that the relation "$w'_{r'}(e_1) = 0$ or $w'_{r'}(e'_1) = 0$" is equivalent to "$w_r(e) = 0$ or $w_r(e') = 0$". Moreover, from the definition of r' we have that $W'_{r'}(u, v) = W_r(u, v)$. Therefore, Relation (12) must hold.

($\Leftarrow$) Given a retiming r' of G' that satisfies constraints (8) through (12), we must show that there exists a retiming r of G such that G_r is well formed and $\Phi(G_r) \leq c$. By construction, a retiming r can be derived from a retiming r' by simply setting $r(u) = r'(u)$ for all $u \in V$. Therefore, for every edge $e \in E$, we have $w'_{r'}(e_1) + w'_{r'}(e_2) = w_r(e)$. From Inequalities (8) and (10) and the equation $w'_{r'}(e_1) + w'_{r'}(e_2) = w_r(e)$, we infer that $w_r(e) \geq 0$, and thus G_r is well-formed. Moreover, in the previous paragraph we have shown that Relation (12) and Relation (5) are equivalent. Thus, from Relation (5) and Lemma 2, we conclude that $\Phi(G_r) \leq c$. □

The following lemma gives an upper bound on the number of registers in a retimed circuit along any simple path or cycle and it is used to replace Relation (11) by a linear programming constraint.

Lemma 5 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be an edge-triggered circuit and let $u, v \in V$ be any pair of vertices in G. For every legal retiming r, we have*

$$W_r(u, v) \leq F, \tag{13}$$

where $F = \max\{W(u, v) + W(v, u) : u, v \in V\}$.

Proof. The vertices u, v lie either on a cycle or on a cycle-free path from the primary inputs to the primary outputs. From [13], we know that retiming leaves the number of registers in a cycle unchanged. Moreover, for input-to-output cycle-free paths, we can impose retiming constraints to ensure that the latency of the computation remains unchanged. Thus, the maximum number of registers over the minimum register-count paths between u and v in G_r can be no greater than either the number of registers in the cycle containing vertices u, v with the minimum register-count in G or the latency of the input-to-output cycle-free path on which vertices u, v lie. Since $W(u, v) + W(v, u)$ denotes the register count of the minimum weighted cycle through (u, v), and F denotes the maximum over all minimum weighted cycles, we conclude that Inequality (13) must hold. □

A corollary of Lemma 5 is that the register count on any edge in a retimed synchronous circuit G cannot exceed F. The following lemma, with the help of Lemma 5, shows that we can replace Relations (11) and (12) by linear inequalities without changing the solutions set of the original constraints.

Lemma 6 *Consider the constraints set of Lemma 4. For every pair of edges $u \xrightarrow{e_1} x_{uv}$ and $x_{uv} \xrightarrow{e_2} v$ in E', let Relation (11) be replaced by the inequality*

$$w'_{r'}(e_2) \leq F \cdot w'_{r'}(e_1) \ , \tag{14}$$

where $F = \max\{W(u, v) + W(v, u) : u, v \in V\}$. Moreover, for every pair of edges $u' \xrightarrow{e_1'} x_{u'u}$, $v \xrightarrow{e_1'} x_{vv'}$ in E' such that $u' \xrightarrow{e} u$, $v \xrightarrow{e'} v'$ in E, $u \rightsquigarrow v$ in G, and $D_{total}(e, e') > c$, let Relation (12) be replaced by the inequality

$$w'_{r'}(e_1) + w'_{r'}(e'_1) \leq W'_{r'}(u, v) + 1 \ . \tag{15}$$

Then the resulting set of constraints is equivalent to the original one.

Proof. We will show that from any solution r for Inequalities (8), (9), and (10) and Relations (11) and (12) we can derive a solution for Inequalities (8), (9), (10), (14), and (15) and vice-versa.

We first show that Inequality (14) implies Relation (11) by a case analysis of $w'_{r'}(e_1)$. If $w'_{r'}(e_1) = 0$, then Inequality (14) implies that $w'_{r'}(e_2) \leq 0$. From Inequality (10) we conclude that $w'_{r'}(e_2) = 0$, and thus Relation (11) holds. If $w'_{r'}(e_1) \geq 1$, then Inequality (14) implies that $w'_{r'}(e_2) \leq F$. From Lemma 5, we have that F is an upper bound on $w'_{r'}(e_2)$, and thus the inequality $w'_{r'}(e_2) \leq F$ does not impose any additional restriction on the value of $w'_{r'}(e_2)$. Thus, Relation (11) holds.

We now show that Inequality (15) implies Relation (12) by a case analysis of $W'_{r'}(u, v)$. For $W'_{r'}(u, v) = 0$, Inequality (15) implies that $w'_{r'}(e_1) + w'_{r'}(e'_1) \leq 1$. Since register counts are integers, from Inequality (8) we infer that either $w'_{r'}(e_1) = 0$ or $w'_{r'}(e'_1) = 0$, and thus Relation (12) holds. For $W'_{r'}(u, v) \geq 1$, Inequality (15) implies that $w'_{r'}(e_1) + w'_{r'}(e'_1) \leq 2$. This constraint can also be derived from Inequality (9), however, and places no additional restrictions on the values of $w'_{r'}(e_1)$ and $w'_{r'}(e'_1)$. Thus, Relation (12) holds.

The above arguments can be reversed in a straightforward manner to show the other direction of the proof. □

We conclude this section with the following theorem which describes the necessary and sufficient conditions of the retiming problem as integer linear programming constraints on the companion graph G'.

Theorem 7 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be an edge-triggered circuit, let $G' = \langle V', E', w' \rangle$ be its companion graph, and let c be a positive real number. There exists a retiming $r : V \to \mathbf{Z}$ such that $\Phi(G_r) \leq c$ if and only if there exists a function $r' : V' \to \mathbf{Z}$ such that for every edge $u \xrightarrow{e_1} x_{uv}$ in E', we have*

$$0 \leq w'(e_1) + r'(x_{uv}) - r'(u) \ ,$$

$$w'(e_1) + r'(x_{uv}) - r'(u) \leq 1 \ ,$$

for every edge $x_{uv} \xrightarrow{e_2} v$ in E', we have

$$0 \leq w'(e_2) + r'(v) - r'(x_{uv}) \ ,$$

for every pair of edges $u \xrightarrow{e_1} x_{uv}$ and $x_{uv} \xrightarrow{e_2} v$ in E', we have

$$w'(e_2) + r'(v) - r'(x_{uv}) \leq F \cdot (w'(e_1) + r'(x_{uv}) - r'(u)) \ ,$$

and for every pair of edges $u' \xrightarrow{e_1} x_{u'u}, v \xrightarrow{e_1'} x_{vv'}$ in E' such that $u' \xrightarrow{e} u$, $v \xrightarrow{e'} v'$ in E, $u \rightsquigarrow v$ in G, and $D_{total}(e, e') > c$, we have
$w'(e_1) + r'(x_{u'u}) - r'(u') + w'(e_1') + r'(x_{vv'}) - r'(v) \leq W'(u, v) + r'(v) - r'(u) + 1$.

Proof. Follows directly from Lemmas 4 and 6 and Equation (3). □

6 Monotonic Programming Formulation

The integer linear programming constraints of Theorem 7 do not appear to have any special structure and thus we need to resort to general ILP solvers to compute a solution. There are restricted versions of the delay model, however, for which the retiming problem can be solved using more efficient techniques. In this section, we give an integer monotonic programming formulation of the retiming problem for the special case of one-way end-extendible circuits. We also show how to obtain an $O(VE)$-size set of reals that includes the minimum period achievable by retiming. The symmetric results hold for one-way begin-extendible circuits.

Lemma 2 can be expressed for one-way end-extendible circuits as follows.

Lemma 8 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be a one-way end-extendible circuit. Moreover, let c be a positive real number, and let $r : V \to \mathbf{Z}$. Then r is a legal retiming of G such that $\Phi(G_r) \leq c$ if and only if for every edge $u \xrightarrow{e} v$ in G, we have*

$$r(u) - r(v) \leq w(e), \tag{16}$$

and for every edge pairs $i, j \in E$ such that $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ and $D_{total}(i, j) > c$, we have

$$W_r(u, v) = 0 \ \Rightarrow \ w_r(j) = 0 \ . \tag{17}$$

Proof. ($\Rightarrow$) Let G_r be a well-formed circuit such that $\Phi(G_r) \leq c$. Since G_r is well formed, all edge-weights w_r in G_r must satisfy $w_r \geq 0$, and thus, from Equation (1) we infer that Inequality (16) holds.

If $D_{total}(i,j) > c$, $W_r(u,v) = 0$, and $w_r(j) \geq 1$, then there exists a path $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ whose delay exceeds c. Since G is one-way end-extendible, the delay of every register-to-register path that begins on or before i, passes through u, and ends at j will exceed $D_{total}(i,j)$. Thus, there exists a register-to-register path whose propagation delay exceeds the clock period c. This conclusion contradicts our assumption that $\Phi(G_r) \leq c$, and therefore Relation (17) must hold.

($\Leftarrow$) By contradiction. Assume that Inequality (16) and Relation (17) are satisfied by a retiming r of G and that $\Phi(G_r) > c$. Inequality (16) and Equation (1) ensure that the retimed circuit is well-formed, and thus the retiming r is legal. The condition $\Phi(G_r) > c$ implies that there exists a register-to-register combinational path $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ such that $D_{total}(i,j) > c$. By the definition of the register-to-register path, we have $w_r(j) \geq 1$. Since $D_{total}(i,j) > c$, the last inequality implies that Relation (17) is violated which is a contradiction. □

The following lemma, with the help of Lemma 5, leads to an integer linear programming formulation of the constraints in Lemma 8.

Lemma 9 *For every pair of edges $i, j \in E$ such that $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ and $D_{total}(i,j) > c$, we have that Relation (17) is equivalent to the inequality*

$$w_r(j) \leq F \cdot W_r(u,v) \ . \tag{18}$$

Proof. Since the register count along any path is nonnegative, Inequality (18) implies that when $W_r(u,v) = 0$ we have $w_r(j) = 0$. Moreover, it implies that when $W_r(u,v) \geq 1$ we have $w_r(j) \leq F$. By Lemma 5, we know that the register count along any path and consequently on any edge in a retimed circuit cannot exceed F. Thus, $w_r(j) \leq F$ implies that there is no restriction on $w_r(j)$. This condition is identical to the condition in Relation (17). □

Inequalities (13) and (18) are illustrated in Figure 4. We want to impose the constraint $w_r(j) = 0$ when $W_r(u,v) = 0$, and to place no restriction on $w_r(j)$ otherwise. The solution space for $w_r(j)$ is denoted by the points in Figure 4. The bound in Lemma 5 implies that the solution space can be restricted to the region enclosed by the horizontal axis and the three lines in the graph. Since $w_r(j)$ is an integer, the solution space corresponds to the integral points in this region.

The following theorem states the retiming problem for circuits that are one-way end-extendible as an integer linear program with $O(E^2)$ constraints.

Theorem 10 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be a one-way end-extendible circuit. Moreover, let c be an arbitrary positive real number, and let $r : V \rightarrow \mathbf{Z}$. Then r is a legal retiming of G such that $\Phi(G_r) \leq c$ if and only if for every edge $u \xrightarrow{e} v$ in G, we have*

$$r(u) - r(v) \leq w(e) \ , \tag{19}$$

and for every pair of edges $i, j \in E$ such that $u_i \xrightarrow{i} u \rightsquigarrow v \xrightarrow{j} v_j$ and $D_{total}(i,j) > c$, we have

$$r(v_j) - r(v) + w(j) \leq F \cdot (r(v) - r(u) + W(u,v)) \ . \tag{20}$$

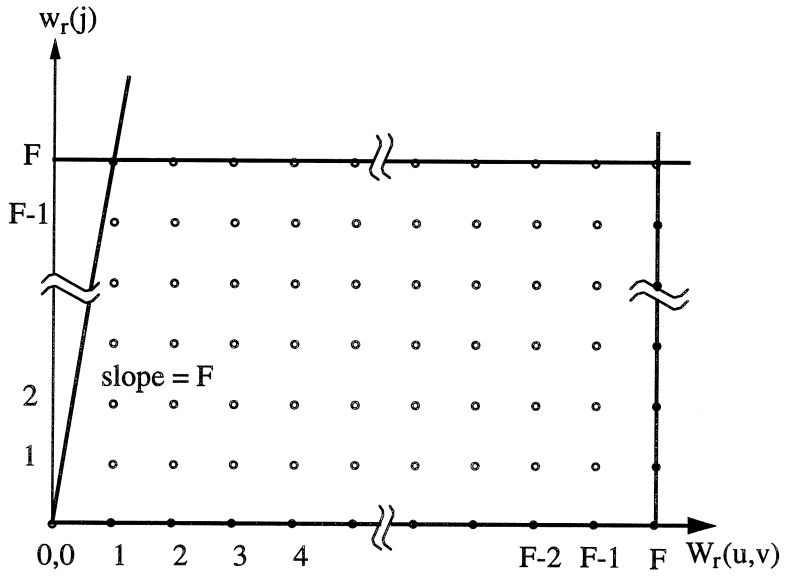

Figure 4: Graphical illustration of Inequalities (13) and (18). The points on the plane denote the solution space for $w_r(j)$. If the problem is feasible, then the region enclosed by the four lines includes an integral solution.

Proof. Immediate from Lemmas 8 and 9 and the definition of $W_r(u, v)$. □

The inequalities in Theorem 10 are in the special form of a *simple integer monotonic programming* problem [9, 18] which is defined as follows.

Problem SIMP *(Simple Integer Monotonic Programming)* Let S be a set of constraints over the unknowns $x_1, x_2, \cdots, x_n$, in which the k^{th} constraint has the form

$$f_k(x_i) \geq g_k(x_1, x_2, \cdots, x_n) \ , \tag{21}$$

where the function f_k is a monotonically increasing in the single unknown x_i and g_k is a monotonically increasing in the unknowns x_j, for $j = 1, 2, \cdots, n$. The simple integer monotonic programming problem is to find a vector $x = \langle x_1, x_2, \cdots, x_n \rangle$ of integers satisfying S, or to determine that no feasible vector exists. □

Theorem 11 *The retiming problem for any one-way end-extendible circuit $G = \langle V, E, d, w, t_b, t_e \rangle$ can be reduced to Problem SIMP.*

Proof. In order to prove the lemma, we must show that Inequalities (19) and (20) from Theorem 10 can be written in the form $f(r(v)) \geq g(r(u), r(v_j))$, where f and g are monotonic functions.

For every edge $u \xrightarrow{e} v \in E$, Inequality (19) can be written as

$$r(v) + w(e) \geq r(u) \tag{22}$$

which has the desired form since both sides of the inequality are monotonic. By rearranging the terms of Inequality (20) we obtain

$$(F + 1) \cdot r(v) + F \cdot W(u, v) \geq F \cdot r(u) + r(v_j) + w(j) \tag{23}$$

MONORELAX (S)
1 **for** $i \leftarrow 1$ **to** n
2 $\quad x_i \leftarrow 0$
3 **while** there exists an unsatisfied constraint in S
4 $\quad$ pick an unsatisfied constraint $f_k(x_i) \geq g_k(x_1, x_2, \cdots, x_n)$
5 $\quad$ **repeat** $x_i \leftarrow x_i + 1$
6 $\quad$ **until** constraint satisfied
7 $\quad S \leftarrow S \cup \{$all constraints with x_i on their r.h.s.$\}$
8 **return** $\langle x_1, x_2, \cdots, x_n \rangle$

Figure 5: Algorithm MONORELAX for solving a simple integer monotonic problem S with unknowns $x_1, x_2, \cdots, x_n$. The procedure returns a solution if the constraints in S can be satisfied.

Since $F > 0$, both sides of the inequality are monotonically increasing and Inequality (23) has the desired form. □

It can be shown that for one-way end-extendible circuits, for each pair of vertices $u, v \in V$, we only need to consider the constraints corresponding to the worst-case fanin edge of u. As a result, the $O(E^2)$ constraints in Theorem 10 can be reduced to $O(VE)$ constraints. This number can be reduced further down to $O(V^2)$, since between any two vertices u and v it suffices to consider the worst-case fanin edge of u and the fanout edges of v that lead to a clock period violation. If the total combinational delay from the worst-case fanin of u to some fanout edges of v exceeds the desired clock period, then from Relation (17) we have that none of these fanout edges should have a register after retiming. Thus, for each pair of vertices u and v, the constraints described by Inequality (18) can be written as a single inequality

$$\sum_{v \xrightarrow{j} v_j} w_r(j) \leq V \cdot F \cdot W_r(u, v) \,. \tag{24}$$

The index j ranges over all fanout edges of v for which $D_{total}(i, j) > c$, where i denotes u's worst-case fanin edge. Inequality (24) is monotonic, since it can be expressed in the following form

$$\sum_{v \xrightarrow{j} v_j} w(j) + \sum_{v \xrightarrow{j} v_j} r(v_j) + V \cdot F \cdot r(u) \leq (V \cdot F + \sum_{v \xrightarrow{j} v_j} 1) \cdot r(v) + V \cdot F \cdot W(u, v) \,. \tag{25}$$

7 An Efficient Retiming Algorithm

In this section, we describe an efficient algorithm for solving the retiming problem for one-way end-extendible circuits. Our algorithm relies on the monotonic programming formulation of the retiming problem described in Section 6.

A feasible solution to a simple integer monotonic problem can be computed using Algorithm MONORELAX which is described in Figure 5. The monotonic constraints for the retiming problem described by Inequalities (22) and (23) have an important property that allows us to solve them by applying Algorithm MONORELAX. Specifically, if $\overline{r}$ is a solution for these inequalities, then $\overline{r} + k$ is also a solution, where k is any integer. Consequently, any solution which results in negative values for $r(v)$ can be shifted up by a constant to yield a nonnegative solution. Moreover, no variable $r(v)$ needs to increase above a minimum value

$\overline{r}(v)$, since by the monotonicity of f and g that minimum value is also a solution. As a result, one can find a minimum nonnegative solution to the retiming problem if a solution exists.

The following lemma shows that if the retiming problem is feasible, then F gives an upper bound for the minimum solution $\overline{r}$.

Lemma 12 *Let $G = \langle V, E, d, w, t_b, t_e \rangle$ be a one-way end-extendible circuit, and let $\overline{r}$ be the minimum nonnegative retiming that achieves the minimum clock period for this circuit. Then for every vertex $v \in V$, we have*

$$\overline{r}(v) \leq F \tag{26}$$

Proof. Let u be the vertex such that $\overline{r}(u) \leq \overline{r}(v)$ for all $v \in V$. This implies that $\overline{r}(u) = 0$. If not, we can subtract $\overline{r}(u)$ from the retiming of all the vertices and still satisfy Inequalities (22) and (23) which contradicts our assumption that $\overline{r}$ is minimum. For the sake of contradiction, let there exist a vertex v with a retiming $\overline{r}(v) > F$. Consider a minimum weighted simple path p from v to u. Adding by parts Inequality (16) along p, we obtain

$$\overline{r}(v) - \overline{r}(u) \leq W(v, u) \ .$$

Since $\overline{r}(u) = 0$, we obtain

$$\begin{aligned} \overline{r}(v) &\leq W(v, u) \\ &\leq F \ , \end{aligned} \tag{27}$$

which contradicts our original assumption. □

Our Algorithm GDMFEAS for the efficient retiming of one-way extendible circuits is described in Fig. 2. The operation of this algorithm is based on the fact that simple integer programming programs can be solved by iterative relaxations [9, 18]. Initially, all constraints corresponding to Inequalities (22) and (23) are inserted into a queue, and all variables $r(v)$ are set to zero. The algorithm proceeds by iteratively removing constraints from the queue. For each violated constraint, it increases the value of the variable on its left-hand side and reinserts into the queue any constraints that are violated as a result of this operation. This iterative scheme continues until all constraints are satisfied or until the value of some variable exceeds F. In the latter case, the algorithm concludes that the specified clock period cannot be achieved by retiming.

Theorem 13 *Given a one-way end-extendible circuit $G = \langle V, E, d, w, t_b, t_e \rangle$ and a real number c, Algorithm* GDMFEAS *determines in $O(V^3 F)$ time a retiming r such that $\Phi(G_r) \leq c$ or that such a retiming does not exist.*

Proof. It can be shown from Theorem 11 and Lemma 12 that Algorithm GDMFEAS computes a unique nonnegative solution for Inequalities (22) and (23) in which every variable $r(v)$ attains its minimum value that does not exceed F [11].

We now show that Algorithm GDMFEAS terminates in $O(V^3 F)$ steps. The parameters D and W required for Inequalities (22) and (23) in Step 1 can be computed in $O(VE + V^2 \lg V)$ time using Johnson's algorithm for all-pairs shortest-paths [5]. The functions $f(r(v))$ and $g(r(u), r(v_j))$ can be computed in $O(1)$ time. Since we have only V variables and no variable can exceed F, the body of the **while** loop is executed $O(VF)$ times. Since

```
GDMFEAS(G,c)
1  Q ← { Inequalities (22) and (23) from Theorem 11}
2  for every vertex v ∈ V
3     r(v) ← 0
4  while Q ≠ ∅
5     remove a constraint "f(r(v)) ≥ g(r(u), r(v_j))" from Q
6     if f(r(v)) < g(r(u), r(v_j))
7        repeat  r(v) ← r(v) + 1
8                if r(v) > F
9                   return fail
10       until f(r(v)) ≥ g(r(u), r(v_j))
11       Q ← Q ∪ {all constraints with r(v) on their r.h.s.}
12 return r
```

Figure 6: Algorithm GDMFEAS for retiming a given circuit G to achieve a specified clock period c.

for each increment of variable $r(v)$, at most $O(V^2)$ constraints with $r(v)$ on their right-hand side are inserted in the queue, Step 11 takes $O(V^2)$ time. Consequently, the total running time of Algorithm GDMFEAS is $O(V^3F)$. □

The optimal retiming problem can be solved in $O(V^3F \lg V)$ steps by using Algorithm GDMFEAS to binary search the set of $O(VE)$ potential clock periods.

8 Experimental Results

We have implemented our retiming algorithms in an efficient and versatile software tool called DELAY that has been developed at Yale University. DELAY has been implemented using the C programming language and integrated into the SIS tools from Berkeley. In order to evaluate the efficiency of our new retiming algorithms and the limitations of previous retiming algorithms that assume simple delay models, we have implemented three algorithms in our tool.

The first implementation is an adaptation of the $O(V^3 \lg V)$-time Bellman-Ford algorithm (BF) from [13] for optimal retiming. We have modified the original algorithm so that the parameters $D_{total}(u, v)$ are used as the critical path delays between u and v instead of $D(u, v)$. Even though the critical delays $D_{total}(u, v)$ are computed under a general delay model, the algorithm still applies the path-breaking approach to satisfy clock period constraints. As a result, this approach, which is optimal for the simple delay model, is still sub-optimal for the general delay model. The purpose of this implementation was to investigate how far from their optimal values are the clock periods of the retimed circuits it generates. The second algorithm (IMP) is the $O(V^3F \lg V)$-time Algorithm GDMFEAS for optimal retiming using integer monotonic programming. Our third implementation (ILP) solves the integer linear programming formulation of the retiming problem described in Section 5. In this implementation, DELAY generates the integer linear programming constraints which are then solved separately by `lp_solve`, a public-domain mixed-integer linear programming solver [1]. In all three implementations, the optimal retiming is computed by a binary search over the set of potential clock periods.

In addition to these three algorithms, we implemented a modified version of the $O(VE \lg V)$

Figure 7: Generation of ill-formed circuits by modified version of the Leiserson-Saxe retiming algorithm. When $d(s \overset{p_1}{\rightsquigarrow} u) > c$ for a given clock period c and $d(s \overset{p_2}{\rightsquigarrow} v) \leq c$, then the algorithm results in a negative register count on the edge $u \rightarrow v$ in the retimed circuit.

algorithm by Leiserson and Saxe so that path delays would be computed based on the general delay model. This modified retiming algorithm, however, can generate ill-formed circuits with negative register counts on some edges when paths have non-monotonic delay. In the remainder of this paragraph we describe this phenomenon, assuming that the reader is familiar with this algorithm as presented in [13]. Let the delay $d(p_1)$ of the combinational path $s \overset{p_1}{\rightsquigarrow} u$ in Figure 7 be greater than the desired clock period c and let $u \rightarrow v$ be an interconnection with no registers. The modified retiming algorithm increments $r(u)$ and attempts to break this long path by inserting a register somewhere on $s \rightsquigarrow u$. If the delay $d(p_2)$ along the path $s \overset{p_2}{\rightsquigarrow} v$ is less than c (which can happen under the general delay model when path delays are non-monotonic), the algorithm does not increment $r(v)$, since the path $s \overset{p_2}{\rightsquigarrow} v$ does not violate the clock period constraint. As a result, we have $r(v) < r(u)$, and edge $u \rightarrow v$ ends up with a negative register count in the retimed circuit.

We experimented with DELAY on the MCNC suite of benchmark circuits by applying the following methodology. We first mapped the test circuits using a version of `lib2.mis2lib` that was augmented to include flip-flops. We then used random numbers to obtain values for t_b and t_e for every interconnection in the circuit, thus emulating the effects of clock skew, load-dependent delays, setup delays and register delays. We considered two different random number distribution for t_b and t_e: a uniform distribution and a gaussian distribution in the range of $[0, 2 \cdot d_{max}]$, where d_{max} was the maximum propagation delay of the circuit components. The standard deviation of the gaussian distribution was $d_{max}/\sqrt{12}$. For both distributions, we applied the following experimental procedure. We first retimed each circuit using TIM [20], a timing package that assumes the simple delay model described in [13]. We then calculated the actual clock period under the general delay model of the optimal circuit generated by TIM, and we compared it with the clock period obtained using the three algorithms BF, IMP, and ILP in DELAY. Our experiments were performed on a SPARC 10 with 64 MB of main memory.

Our experimental results are summarized in the tables of Figures 8 and 9 for the uniform and the gaussian distribution, respectively. In each table, the first and the second column give the name and the size of the test circuits. The third column ($\Phi(G_r)$ (simple)) gives the clock period of the optimally retimed circuits using TIM under the simple delay model. The actual clock periods of the retimed circuits under the general delay model are given in the fourth column ($\Phi(G_r)$(general)). The fifth, sixth and seventh columns give the clock period of optimally retimed circuits under a general delay model using algorithms ILP, IMP, and BF, respectively.

Our experiments show that algorithms which are exact for simpler delay models demonstrate varying degrees of sub-optimality when optimizing circuits under the general delay model. TIM employs the Leiserson-Saxe algorithm for the simple delay model given in [13] and does not use any information about the general delay model. As a result, circuits optimized by TIM have very large clock periods under the general delay model. Similarly, Algorithm BF generates sub-optimal clock periods. Since this algorithm computes path delays using the general delay model, it yields better clock periods than those generated by TIM. IMP is an exact algorithm for retiming circuits with only one-way extendible

Circuit	circuit size	optimal G_r with TIM		optimal G_r with DELAY		
	(no. of gates)	$\Phi(G_r)$ (simple)	$\Phi(G_r)$ (general)	ILP $\Phi(G_r)$ (general)	IMP $\Phi(G_r)$ (general)	BF $\Phi(G_r)$ (general)
train4	12	6.87	13.40	11.23	11.23	12.31
s8	17	7.48	14.78	13.53	14.09	14.18
beecount	26	8.90	18.44	14.75	15.00	16.52
bbara	37	11.31	20.18	18.29	18.29	18.29
dk14	43	11.85	23.60	12.34	14.72	19.86
sse	52	12.91	23.88	23.03	23.09	23.28
mark1	59	11.36	21.61	19.53	19.53	21.69

Figure 8: A comparison of clock periods of optimally retimed circuits generated by the three retiming algorithms BF, IMP, and ILP implemented in DELAY with those generated by TIM. The delay values for t_b and t_e were assigned from a uniform distribution.

paths. This delay model is more sophisticated than those assumed by TIM or BF, and thus, even though IMP generates circuits with sub-optimal clock periods under the general delay model, it gives better results than those generated by BF. Among the three algorithms in DELAY, ILP is exact for our general delay model and generates the optimal retimed circuit with the smallest clock period.

Our experiments also indicate that the distribution of the random numbers makes a difference in the results generated by ILP, IMP and BF. Under a gaussian distribution, the random numbers for t_b and t_e are close to each other which leads to a small number of non-monotonic paths in the circuit. Under a uniform distribution, however, the random numbers for t_b and t_e can take values far from each other, thus increasing the number of non-monotonic paths. For this reason, when with the results obtained under the gaussian distribution, the optimal clock periods computed by IMP and BF are farther away from the optimal clock periods generated by ILP under the uniform distribution.

In addition to experimenting with the effectiveness of our implementations, we evaluated their efficiency in terms of their running times. A comparative graph of the running times is given in Figure 10. ILP is considerably slower than the other retiming algorithms. The running times of IMP and BF are comparable, while TIM is faster than both. During the binary search performed in ILP, retimings for feasible clock periods can be computed quickly. For detecting infeasible clock periods, however, ILP takes a long time since it searches the entire solution space of the constraints. As a result, ILP takes significantly more time to compute a solution compared to IMP, BF and TIM. We believe that better running times for ILP can be obtained by using more sophisticated software packages for solving integer linear programs, in place of the public domain software `lp_solve`. Our experiments with ILP were restricted in circuit size by the large memory requirements of our implementation. The $O(E^2)$ constraints that need to be generated for this formulation blow up very quickly with increasing circuit size. We were able to run IMP, BF and TIM on significantly larger circuits due to their smaller memory requirements.

In conclusion, IMP offers a good compromise in terms of speed and optimality of clock period. Furthermore, the IMP formulation has more potential than what is evident from our empirical results. Our IMP implementation generates clock period constraints assuming

Circuit name	circuit size	optimal G_r with TIM		optimal G_r with DELAY		
	(no. of gates)	$\Phi(G_r)$ (simple)	$\Phi(G_r)$ (general)	ILP $\Phi(G_r)$ (general)	IMP $\Phi(G_r)$ (general)	BF $\Phi(G_r)$ (general)
train4	12	6.87	10.61	8.32	8.89	9.95
s8	17	7.48	12.25	11.20	11.20	11.31
beecount	26	8.90	15.58	12.06	12.75	12.75
bbara	37	11.31	17.93	15.25	15.25	15.25
dk14	43	11.85	21.88	18.59	18.59	19.46
sse	52	12.91	20.77	18.89	18.89	19.10
mark1	59	11.36	19.13	16.87	16.87	16.97

Figure 9: A comparison of the clock periods of optimally retimed circuits generated by the three retiming algorithms BF, IMP, and ILP implemented in DELAY with those generated by TIM. The delay values for t_b and t_e were assigned from a gaussian distribution.

that all paths in the given circuit are end-extendible. A preprocessing step that classifies paths or vertex-pairs as begin-extendible or end-extendible would, however, greatly improve the effectiveness of IMP without affecting its running time. We believe that this augmentation to IMP would result in retimed circuits with clock periods very close to the clock periods of the optimally retimed circuits generated by ILP, but in a significantly shorter time.

9 Conclusion and Future Work

In this paper we have investigated the retiming problem for edge-triggered circuits under a general delay model that includes the timing effects of load-dependent delays, register delays, interconnect delays, and clock skew. For the general retiming problem, we have given an integer linear programming formulation. In this formulation, however, the constraints are not in any special form and we need to resort to general algorithms for integer linear programming in order to solve them. For a restricted version of our general delay model which includes one-way extendible circuits, we have presented an efficient retiming algorithm whose running time is a polynomial in the number of circuit components. The design of this efficient algorithm has relied on a monotonic programming formulation of the retiming problem. We have incorporated our retiming algorithms into DELAY, a versatile and efficient tool for timing optimization of synchronous circuits, and presented empirical results demonstrating their effectiveness.

Even though the delay model we have considered is more general than other models previously considered in the literature, it still has its shortcomings. Most notably, this model does not take into account load-dependent delays due to multiple fanouts. When the output of a gate u fans out to several other gates, then the propagation delay of a path p that goes through u depends on the load presented by the other gates, even though these gates are not on p. Thus, as retiming relocates registers, the propagation delay along p can change even if the registers in the beginning and the end of p are not relocated. In this case, the retiming problem is complicated by the fact that it may be possible to fix a timing violation by relocating registers adjacent to a violating path instead of registers in

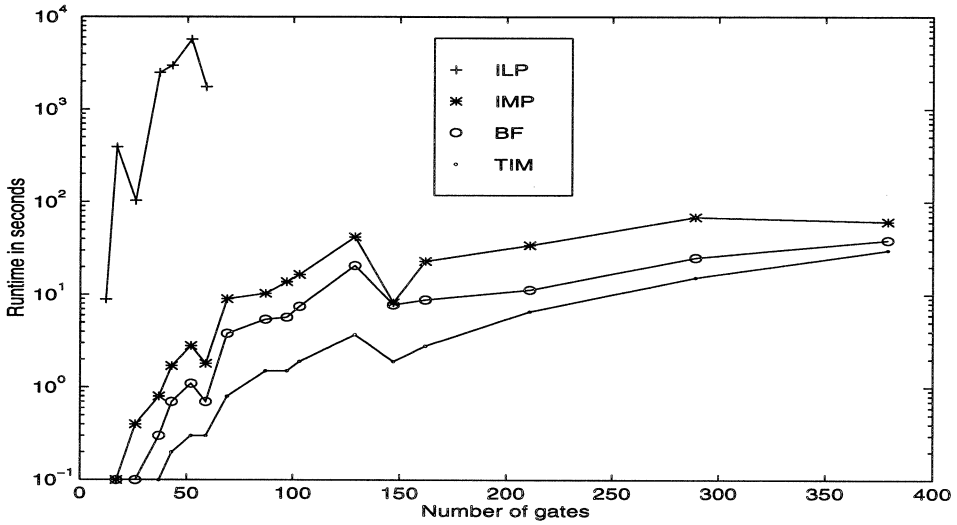

Figure 10: Semilog plot of runtimes (in CPU seconds) against circuit size (in number of gates) for algorithms ILP, IMP, BF, and the retiming algorithm in TIM.

the beginning or the end of the path. Even though we can express this retiming problem in terms of register-to-register constraints that are amenable to iterative relaxation schemes, we cannot guarantee that these schemes will always succeed in finding a solution when the problem is feasible.

We are currently working on improving the performance and applicability of DELAY. Our current implementations generate and solve an exhaustive set of constraints whose size can be significantly reduced by eliminating redundant constraints. We are planning to improve the performance of our tool by reducing the number of constraints it generates for a given clock period. We are also developing algorithms for retiming level-clocked circuits with general delay models. The main challenge in level-clocked retiming under general delay models is that timing violations can be fixed by moving latches *along* paths in addition to moving latches *around* paths. This phenomenon appears to make the problem intractable, and we are investigating more restricted models where we assume that violating paths can be fixed only by moving the latches on either end of the path. Our preliminary investigations reveal that it is possible to express the retiming problem as an integer linear program under this restricted condition. In future research, we will also extend our retiming algorithms to handle precharged gate circuits with general delay models.

Acknowledgments

We would like to thank Jean-Marc Delosme of Yale for helping us with several VLSI modeling issues. We would also like to thank Carl Ebeling of the University of Washington and Charles Leiserson of MIT for several helpful discussions.

References

[1] M. Berkelaar. `lp_solve`: A mixed-integer linear programming solver. Available by anonymous ftp from `ftp://ftp.es.ele.tue.nl/pub/lp_solve`.

[2] S. Chakradhar and S. Dey. Resynthesis and retiming for optimum partial scan. In *Proceedings of the 31st ACM/IEEE Design Automation Conference*, pages 87–93, June 1994.

[3] L. F. Chao and E. H. Sha. Retiming and clock skew for synchronous systems. In *Proceedings of International Symposium on Circuits and Systems*, pages 283–286, June 1994.

[4] F. Commoner, A. W. Holt, S. Even, and A. Pnueli. Marked directed graphs. *Journal of Computer and System Sciences*, 5:511–523, 1971.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, MIT Press, 1990.

[6] S. Dey and S. Chakradhar. Retiming sequential circuits to enhance testability. In *Proc. of the 12th IEEE VLSI Test Symposium*, pages 28–33, April 1994.

[7] J. P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, 39(7):945–951, July 1990.

[8] D. V. Heinbuch. *CMOS3 Cell Library.* Addison Wesley, 1988.

[9] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. In *Advanced Research in VLSI and Parallel Systems: Proc. of the 1992 Brown/MIT Conference*, pages 245–264. MIT Press, March 1992.

[10] K. N. Lalgudi and M. C. Papaefthymiou. DELAY: An efficient tool for retiming with realistic delay modeling. In *Proceedings of the 32th ACM/IEEE Design Automation Conference*, pages 304–309, June 1995.

[11] K. N. Lalgudi and M. C. Papaefthymiou. Efficient retiming under a general delay model. In *Advanced Research in VLSI: Proc. of the 1995 Chapel Hill conference*, pages 368–382. IEEE Computer Society Press, March 1995.

[12] C. E. Leiserson and J. B. Saxe. Optimizing synchronous systems. *Journal of VLSI and Computer Systems*, 1(1):41–67, 1983.

[13] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):1–27, 1991.

[14] B. Lockyear and C. Ebeling. Optimal retiming of multi-phase, level-clocked circuits. In *Advanced Research in VLSI and Parallel Systems: Proc. of the 1992 Brown/MIT Conference.* MIT Press, March 1992.

[15] B. Lockyear and C. Ebeling. The practical application of retiming to the design of high performance systems. In *Digest of Technical Papers of the 1993 IEEE International Conference on CAD*, pages 288–295, November 1993.

[16] G. De Micheli. Synchronous logic synthesis: algorithms for cycle-time minimization. *IEEE Transactions on Computer-Aided Design*, 10:63–73, January 1991.

[17] J. Monteiro, S. Devadas, and A. Ghosh. Retiming sequential circuits for low power. In *Digest of Technical Papers of the 1993 IEEE International Conference on CAD*, pages 398–402, November 1993.

[18] M. C. Papaefthymiou. *A Timing Analysis and Optimization System for Level-Clocked Circuitry.* PhD thesis, Massachusetts Institute of Technology, September 1993. Available as MIT/LCS/TR-605.

[19] M. C. Papaefthymiou. Understanding retiming through maximum average-delay cycles. *Mathematical Systems Theory*, 27:65–84, 1994.

[20] M. C. Papaefthymiou and K. H. Randall. TIM: a timing package for two-phase, level-clocked circuitry. In *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pages 497–502, June 1993.

[21] T. Soyata and E. Friedman. Retiming with non-zero clock skew, variable register and interconnect delay. In *Digest of Technical Papers of the 1993 IEEE International Conference on CAD*, pages 234–241, November 1994.

[22] T. Soyata, E. Friedman, and J. Mulligan. Integration of clock skew and register delays into a retiming algorithm. In *Proceedings of International Symposium on Circuits and Systems*, pages 1483–1486, May 1993.